ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

Факультет компьютерных наук
Образовательная программа бакалавриата «Программная инженерия»

СОГЛАСОВАНО Научный руководитель, профессор департамента программной инженерии факультета компьютерных наук канд. техн. наук	УТВЕРЖДАЮ Академический руководитель образовательной программы «Программная инженерия» профессор департамента программной инженерии, канд. техн. наук
_____ С.М. Авдошин «___» _____ 2020 г.	_____ В.В. Шилов «___» _____ 2020 г.

Подп. и дата	
Инв. № дубл.	
Взам. инв. №	
Подп. и дата	
Инв. № подл	

ПРОГРАММА КЛАССИФИКАЦИИ КОМПЬЮТЕРНЫХ АТАК ПО НАБОРУ ДАННЫХ ISCX BOTNET

Руководство программиста

ЛИСТ УТВЕРЖДЕНИЯ

RU.17701729.02.13-01 33 01-1-ЛУ

Исполнитель
студент группы БПИ193
_____ /Е.А. Гриценко /
«____»_____ 2020 г.

Москва 2020

УТВЕРЖДЕН
RU.17701729.02.13 -01 33 01-1-ЛУ

Подп. и дата	
Инв. № дубл.	
Взам. инв. №	
Подп. и дата	
Инв. № подл	

ПРОГРАММА КЛАССИФИКАЦИИ КОМПЬЮТЕРНЫХ АТАК ПО НАБОРУ ДАННЫХ ISCX BOTNET

Руководство программиста

RU.17701729.02.13-01 33 01-1

Листов 18

Москва 2020

ОГЛАВЛЕНИЕ

1. НАЗНАЧЕНИЕ И УСЛОВИЯ ВЫОЛНЕНИЯ ПРОГРАММЫ

1.1. Назначение программы и структура

ПО состоит их двух частей программы BotnetDetector и библиотеки классов LibBtntDtct. Программа обладает консольным интерфейсом, описанным в руководстве оператору. В данном документе рассматривается функционал библиотеки классов. Библиотека классов предназначена для захвата информации о сетвых потоков из трафика, его дампа, классифкации этих потоков и создания классификаторов, которые могут быть использованы программой.

1.2. Зависимости.

Зависимости данного ПО описаны в главе 2.1 руководства оператора.

1.3. Вычислительная мощность и память.

Требования к вычислительной мощности и пасяти опсаны в главе 2.4 руководства оператора.

3. ХАРАКТЕРИСТИКИ ПРОГРАММЫ, ВХОДНЫЕ И ВЫХОДНЫЕ ДАННЫЕ

Библиотека классов представляет собой файл LibBtntDtct.dll; библиотеки-зависимости SharpPcap 5.1.0, PacketDotNet 1.0.3, System.Runtime.CompilerServicies.Unsafe 4.6.0 поставляются с ней в архиве и имеют расширение .dll. Библиотека написана на языке C# версии 7.0 и использует .NET Framework 4.7. Публичные классы библиотеки раположеы в пространстве имён LibBtntDtct и имеют следующий состав: AbstractClassifier, ClassifyingFlowAdapter, Flow, FlowAdapter, FlowClass, FlowPacket, FlowReader, InbuiltClassifiers, InbuiltClassifiers.Default, LiveCaptureFlowAdapter, PcapFlowAdapter. Ниже содержится подробное описание их и каждого из предоставляемых публичных или защищённых (protected) членов.

3.1. AbstractClassifier

Публичный абстрактный класс для всех классификаторов. Его корректная реализация позволяет использовать собственные классификаторы в программе при помощи задания опциии “--classifier” способом, указанным в руководстве оператора. Члены:

3.1.1. public abstract IReadOnlyList<FlowClass> FlowClasses { get; }

Список всех классов, в которые данный классификатор классифицирует потоки. Для корректной загрузки классификатора программой и корректного исполнения классификатора необходимо, чтобы этот список и классы, на которые он ссылается, не изменялись после инициализации классификации, поэтому рекомендуется задавать этот список в конструкторе и более не менять. Не допустимо значение null. Не может быть пустым. Классы потоков, на которые ссылаются список не могут иметь значение null. Имена классов обязаны быть уникальными, их ClassListID должен совпадать.

3.1.2. public abstract FlowClass Classify(Flow f)

Функция классификации, принимающаая на вход информацию о потоке и обязанная вернуть класс. Возвращаемый класс должен быть одним из классов списка FlowClasses.

3.1.3. public FlowClass ValidateClassify(Flow f)

Обёртка над функцией Classify. Выполняет её для передаваемого потока и проверяет, является ли возвращённый ею класс корректным. В случае некорректности выбрасывает исключение.

3.1.4. public static bool ValidnessOf(Type t)

Проверяет, является ли передаваемый тип валидным классификатором. Для этого тип должен быть классом, унаследованным от класса AbstractClassifier и у него должен быть публичный конструктор без параметров.

3.2. FlowClass : IEquatable<FlowClass>

Класс, представляющий класс сетевого потока.

3.2.1. protected static long maxClassID

Переменная, указывающая максимальный ClassListID из всех предыдущих созданных наборов экземпляров классов. Используется для присуждения новым экземпляров новых незанятых ClassListID.

3.2.2. public virtual string Name { get; protected set; }

Представляет имя класса. Имена должны быть уникальными в пределах одного разбиения на классы, проверки в библиотеке на уникальность имён не производится.

3.2.3. protected readonly int id

id класса в списке классов. Всегда должна ему соответствовать. Должна быть уникальным в пределах одного разбиения на классы.

3.2.4. public virtual int Id => id;

Предоставляет публичный доступ на чтение к ID класса.

3.2.5. public virtual long ClassListID { get; protected set; }

ID разбиения на классы, к которому принадлежит этот класс. Все используемые разбиения на классы, если их последовательности не равны, должны иметь уникальное значение этого свойства. Для всех классов каждого разбиения это значение должно быть одинаково.

3.2.6. public virtual IReadOnlyList<FlowClass> ClassList { get; protected set; }

Список всех классов, разбиения на классы, к которому принадлежит этот класс. Переменная id должна соответствовать индексу класса в этом списке.

3.2.7. public bool Equals(FlowClass other)

Проверяет, эквивалентен ли этот класс другому классу на основании ID класса и ID разбиения.

3.2.8. public bool OfSameClassGroup(FlowClass other)

Проверяет, принадлежит ли этот класс тому же, что и другой, разбиению на основании ID разбиения.

3.2.9. public bool DeepOfSameClassGroup(FlowClass other)

Проверяет, принадлежит ли этот класс тому же, что и другой, сравнивая поименённо все классы этого разбиения и все классы другого разбиения.

3.2.10. public bool TryReduce(FlowClass other)

Если разбиения на классы для этого и другого класса являются одинаковыми последовательностями, то пытается присвоить всем классам разбиения на классы, к которому принадлежит другой класс, свой ID. Возвращает true в случае успеха, иначе false. Это позволяет проверять равенство классов и принадлежность к другому по-факту такому же разбиению по индексам.

3.2.11. public override bool Equals(object obj)

Проверяет равенство этого класса объекту, а именно — проверяет, является ли другой объект классом типа FlowClass, и если является, возвращает значение функции (из пункта 2.7.) bool Equals(FlowClass other), иначе возвращает false.

3.2.12 protected FlowClass(string name, int id, long classListId)

Конструктор. Присваивает Name, id и ClassListID этого класса передаваемые параметры.

3.2.13. public static FlowClass[] Create(params string[] arr)

Возвращает разбиение на классы с именами из массива. Имена должны быть уникальными, и их уникальность в конструкторе не проверяется.

3.3. Flow

Класс, предоставляющий информацию о сетевом потоке.

3.3.1. public const string timePattern = "dd.MM.yyyy HH:mm:ss.ffffff";

Формат даты и времни, используемый для строкового представления потока.

3.3.2. public DateTime Start { get; set; }

Представляет дату и время начала сетевого потока.

3.3.3. public DateTime End { get; set; }

Представляет дату и время конца сетевого потока.

3.3.4. public IPAddress Source { get; set; }

Представляет IP отправителя.

3.3.5. public ushort SPort { get; set; }

Представляет порт отправителя. Для протоколов, не подразумевающих наличие портов, это свойство должно устанавливаться в 0.

3.3.6. public IPAddress Destination { get; set; }

Представляет IP адрес получателя.

3.3.7. public ushort DPort { get; set; }

Представляет порт получателя. Для протоколов, не подразумевающих наличие портов, это свойство должно устанавливаться в 0.

3.3.8. public int ProtocolID { get; set; }

Представляет номер протокола потока.

3.3.9. public int PacketsCount { get; set; }

Представляет число пакетов.

3.3.10. public int OutgoingPackets { get; set; }

Представляет число исходящих пакетов.

3.3.11. public long OctetsCount { get; set; }

Представляет общее число октет потока.

3.3.12. public long PayloadOctetsCount { get; set; }

Представляет общее число октет в Payload-е пакетов.

3.3.13. public override string ToString()

Возвращает строковое представление этого потока, соответствующее формату, описанному в пункте 1. главы 4.1.2. ТЗ. Для представления значений использует инвариантную культуру.

3.3.14. protected static int Parse(string[] arr, int i, Flow at)

Парсит указанный массив, начиная с 0-ого индекса в свойства в порядке их описания, тримируя каждое из начений (значения в массиве не меняются). Сохраняет данные в параметр at. Возвращает число 11 — следующий индекс после того, на котором завершился парсинг. Для парсинга используется инвариантная культура.

3.3.15. public static void Parse(string s, Flow at)

Парсит строку, соответствующую формату, описанному в пункте 1. главы 4.1.2. ТЗ в параметр at.

3.3.16. public static Flow Parse(string s)

Парсит строку, соответствующую формату, описанному в пункте 1. главы 4.1.2. ТЗ, и возвращает результат парсинга.

3.4. FlowReader

Класс, предназначенный для чтения информации о сетевых потоках из потока формата, описанного в пункте 3 главы 4.1.2. ТЗ.

3.4.1. protected StreamReader reader

Адаптер, на котором базируется этот класс.

3.4.2. public bool EOF => reader.EndOfStream;

Индикатор конца файла или потока.

3.4.3. public Flow ReadFlow()

Читает следующую строку с информацией о потоке и возвращает результат парсинга.

3.4.4. public void ReadFlow(Flow at)

Читает следующую строку с информацией о потоке и записывает результат парсинга в параметр at.

3.4.5. public FlowReader(Stream s)

Конструктор. В качестве параметра указывается поток, из которого читать информацию.

3.4.6. public FlowReader(Stream s, bool skipFirstLine)

Конструктор. В качестве параметра указывается поток, из которого читать информацию, а также параметр, указывающий, пропускать ли первую линию не читая, или нет.

3.5. FlowPacket

Класс, представляющий информацию о пакете, которая в дальнейшем может быть использована для обработки сетевых потоков.

3.5.1. public const int ARP = 470;

Тип протокола, присуждаемый ARP-потокам.

3.5.2. public DateTime Timestamp { get; set; }

Дата и время получения этого пакета.

3.5.3 public IPAddress Source { get; set; }

Представляет IP отправителя.

3.5.4 public ushort SPort { get; set; }

Представляет порт отправителя. Для протоколов, не подразумевающих наличие портов, это свойство должно устанавливаться в 0.

3.5.5. public IPAddress Destination { get; set; }

Представляет IP адрес получателя.

3.5.6. public ushort DPort { get; set; }

Представляет порт получателя. Для протоколов, не подразумевающих наличие портов, это свойство должно устанавливаться в 0.

3.5.7. public int ProtocolID { get; set; }

Представляет номер протокола потока.

3.5.8 public int OctetsCount { get; set; }

Число октет в данном пакете.

3.5.9. public bool HasPayload { get; set; }

Имеет ли этот пакет payload.

3.5.10. public int PayloadLength { get; set; }

Длинна payload-а.

3.5.11. public static void ParseEthernetPacket(EthernetPacket what, out FlowPacket at, bool ignoreMinorErrors = false)

Парсит пакет типа PacketDotNet.EthernetPacket в параметр FlowPacket at. Пареметр ignoreMinorErrors указывает, игнорировать ли незначительные ошибки парсинга (например, ошибку установления длины payload-а).

3.5.12. public static bool TryParseEthernetPacket(EthernetPacket what, out FlowPacket at, bool ignoreMinorErrors = false)

Пытается парсить пакет методом (5.11). В случае успеха, возвращает true, иначе false.

3.6. FlowAdapter

Класс, используемый для выявления сетевых потоков из последовательно добавляемых в него пакетов.

3.6.1. public event Action<Flow> FlowSpawned

Вызывается, когда был вызван новый поток.

3.6.2. public event Action<Flow> FlowDead

Вызывается, когда поток убирается из листинга.

3.6.3. public event Action<Flow, FlowPacket> PacketAppened

Вызывается, когда пакет был добавлен к потоку.

3.6.4. public TimeSpan TimeWindow { get; set; }

Временное окно, по истечении которого все потоки убираются с листинга.

3.6.5. public DateTime LastDate { get; set; }

Дата последнего добавленного пакета.

3.6.7. public List<Flow> Flows { get; set; }

Список текущих потоков.

3.6.8. public Flow SearchFlow(FlowPacket fp)

Ищет поток, которому принадлежит этот пакет. Если не находит возвращает null, если же находит только один, возвращает его, если находит более одного потока, вызывает исключение.

3.6.9. public virtual void KillFlows()

Удаляет все потоки из листинга.

3.6.10. public void ConvertToFlow(FlowPacket fp, Flow flow)

Инициализирует указанный класс потока значениями указанного пакета.

3.6.11. public virtual void ReadPacket(EthernetPacket ethPacket, DateTime packetTime, bool ignoreMinorErrors = true)

Чиатет следующий пакет из аргумента ethPacket и изменяет на его основе текущие потоки. По мимо этого также пердаётся дата и время захвата этого пакета и то, должен ли парсер пакета игнорировать незначительные ошибки (парсер представлен методами 3.5.11 и 3.5.12).

3.6.12. public FlowAdapter()

Конструктор. Инициализирует TimeWindow значением 60 и LastDate значеним DateTime.MaxValue.

3.7. ClassifyingFlowAdapter : FlowAdapter

Класс, используемый для выявления сетевых потоков из последовательно добавляемых внего пакетов и их классификации.

3.7.1. public AbstractClassifier FlowClassifier { get; protected set; }

Классификатор, которым производится классификация.

3.7.2. public event Action<Flow, FlowClass> FlowClassified

Вызывается, когда поток был классифицирован.

3.7.3. public override void KillFlows()

Классифицирует и удаляет все потоки из листинга.

3.7.4. public ClassifyingFlowAdapter(AbstractClassifier classifier)

Конструктор. Приминимает в качестве аргумента классификатор, которым необходимо производить классификацию.

3.8. LiveCaptureFlowAdapter : FlowAdapter

Класс, используемый для выявления сетевых потоков из трафика «живого» устройства.

3.8.1. public long Total { get; set; }

Представляет общее количество принятых с устройства пакетов.

3.8.2. public long Errors { get; set; }

Представляет общее количество пакетов, при парсинге которых произошли ошибки.

3.8.3. public long Dropped { get; set; }

Предстаяляет число пропущенных, не дабавленных в очередь на обработку пакетов в силу её переполненности.

3.8.4. public int QueudPackets => captures.Count;

Возвращает число находящихся в очереди на обработку пакетов.

3.8.5. public int MaxQueueSize { get; set; } = 1000000;

Представляет максимальное допустимое количество находящихся в очереди на обработку пакетов.

3.8.6. public void StopCapture()

Останавливает захват пакетов с устройства и обрабатывает все оставшиеся в очереди пакеты.

3.8.7. public void StartCapture()

Начинает захват и обработку пакетов с устройства.

3.8.8. public LiveCaptureFlowAdapter(ICaptureDevice device)

Конструктор. Принимает на вход представление устройства типа SharpPcap.ICaptureDevice: устройство, прослушивание и обработка пакетов которого должна производиться.

3.8.9. public LiveCaptureFlowAdapter(string deviceName)

Конструктор. Принимает на вход имя устройства, прослушивание и обработка пакетов каторого должна производиться. Ищет в списке устройств, сообщаемом SharpPcap свойством CaptureDeviceList.Instance, устройство с указанным именм. Как только находит, инициализирует класс на основании используемого устройства и прекращает поиск. В случае, если устройство с таковым именем не найдено, выбрасывает исключение.

3.9. PcapFlowAdapter : FlowAdapter, Idisposable

Класс, используемый для выявления сетевых потоков из PCAP-дампа трафика.

3.9.1. public enum Options

Перечисление опций обработки ошибок.

3.9.1.1. Strict

При Strict, при парсинге пакета не игнорируются никакие ошибки парсинга.

3.9.1.2.IgnoreMinorErrors

При IgnoreMinorErrors, при парсинге игнорируются минорные ошибки парсинга (описание в пункте 6.11.)

3.9.1.3.IgnoreMajorErrors

При IgnoreMajorErrors, при парсинге игнорируются минорные ошибки парсинга (описание в пункте 6.11) и подавляются все исключения.

3.9.2. public CaptureFileReaderDevice ReaderDevice { get; private set; }

Устройство (представление типа SharpPcap.LibPcap.CaptureFileReaderDevice), используемое для чтения пакетов.

3.9.3. public Options Option { get; set; }

Свойство, определяющее политику управления ошибками парсинга типа перичисления, описанного в пункте 9.1. Поведение при различных значениях свойства описано там же.

3.9.4. public int ReadNextPcapPacket()

Осуществляет чтение следующего пакета из предоставленного устройства и его передачу в метод ReadPacket базового класса.

3.9.5. public void Dispose()

Реализует интерфейс IDisposable. Закрывает устройство.

3.9.6. public PcapFlowAdapter(string filePath)

Конструктор, принимающий на вход путь к PCAP-файлу, с которого будет производиться чтение пакетов.

3.10. InbuiltClassifiers

Класс, в котором осуществляется хранение встроенных классификаторов.

3.11. InbuiltClassifiers.Default : AbstractClasssifier

Бинарный классификатор, классифицирующий потоки как нормальные или как botnet-потоки.

3.11.1. public override IReadOnlyList<FlowClass> FlowClasses => flowClasses;

Возвращает список классов разбиения: класс с именем “botnet” и класс с именем “normal”.

3.11.2. public double GetFlowAPL(Flow f)

Возвращает среднюю длину payload-а для переданного потока.

3.11.3. public double GetFlowBS(Flow f)

Возвращает среднюю скорость потока, выражаемую количеством байт в секунду. Если время начала и конца совпадает, то считается, что длительность составляет 0.01 секунды.

3.11.4. public double GetFlowIOPR(Flow f)

Возвращает отношение числа исходящик пакетов к общему их числу.

3.11.5. public double GetFlowDuration(Flow f)

Возвращает длительность потока. Если время начала и конца совпадает, то считается, что длительность составляет 0.01 секунды.

3.11.6. public override FlowClass Classify(Flow f)

Классифицирует поток, как normal или как botnet, используя дерево решений, описанное в пояснительной записке в главах 3.9 и 3.10. Решение основывается на значениях APL, BS, IOPR, Duration, рассчитывваемых функциями 3.11.2—3.11.5 соответственно. Максимальная глубина дерева решений — 16. Дерево представлено в виде автосгенерированного C#-кода, и потому при выполнении компилируется непосредственно в исполняемый код. Поэтому классификация должна происходить относительно быстро.

4. ОБРАЩЕНИЕ К ПРОГРАММЕ

Обращение к библиотеке классов осуществляется посредством её добавления как зависимости проекта в средах разработки, поддерживающих это (Microsoft Visual Studio, Monodevelop). Также возможно динамическое подключение библиотеки в ходе выполнения программы, описание динамического подключение библиотек находится в документации Microsoft по .NET Framework в разделе “Application domains and assemblies”.

Для того, чтобы разработанный сторонний классификатор (все поддерживаемые сторонние классификаторы должны быть унаследованы от AbstractClassifier) был использован при запуске программой, необходимо соблюсти изложенные в описании классов AbstractClassifier и FlowClass требования, скомпилировать библиотеку с классификатором в DLL-файл, передать путь к ней и полное имя типа классификатора через опцию “--classifier” способом, определённым в руководстве оператора в главе 3.3.

5. СООБЩЕНИЯ

Методы библиотеки классов не передают сообщения в консоль или камим-либо иным образом.

6. СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. «Программа классификации компьютерных атак по набору данных ISCX Botnet». Руководство оператора;

2. «Программа классификации компьютерных атак по набору данных ISCX Botnet». Техническое задание;

3. «Программа классификации компьютерных атак по набору данных ISCX Botnet». Пояснительная записка.

ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ

Изм.	Номера листов (страниц)				Всего листов (страниц) в документе	№ документа	Входящий № сопроводительного документа и дата	Подпись	Дата
	измененных	замененных	новых	аннулированных					